PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06T 11/00

A1

(11) International Publication Number: WO 97/42603

(43) International Publication Date: 13 November 1997 (13.11.97)

(21) International Application Number: PCT/EP97/02229

(22) International Filing Date: 30 April 1997 (30.04.97)

(30) Priority Data:
08/642,395 3 May 1996 (03.05.96) US

(71) Applicant: THE HARLEQUIN GROUP LTD. [GB/GB]; Barrington Hall, Barrington, Cambridge CB2 5RG (GB).

(72) Inventors: EARL, David, John; 31 Hinton Road, Fulbourn, Cambridge CB1 5DZ (GB). REVIE, William, Craig; 4 Willingham Road, Over, Cambridge CB4 5PD (GB).

(74) Agents: LIESEGANG, Roland et al.; Boehmert & Boehmert, Franz-Joseph-Strasse 38, D-80801 München (DE).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

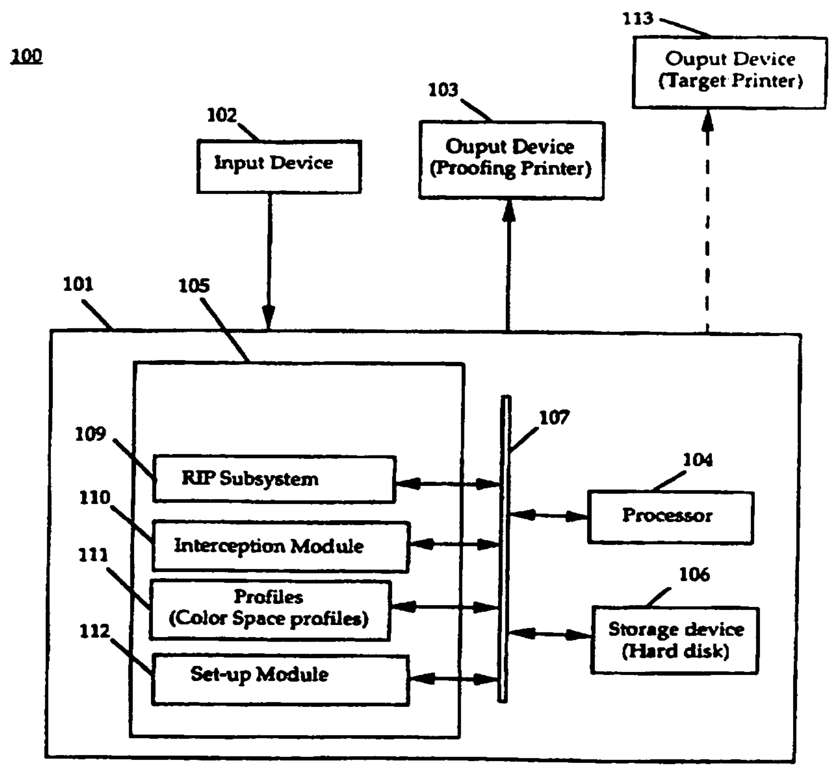
Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR PROCESSING A PAGE DESCRIPTION IN AN IMAGING SYSTEM

(57) Abstract

A system for producing an image from a page description prepared for a target output device includes an input module for providing a page description, memory for storing processing and data modules, a processor operatively coupled to the input module and to the memory for processing the page description using the stored processing and data modules and an output device for forming a physical manifestation of the processed page description. The processing modules and data module provide for the selection of an output device for printing. When the target output device is selected, the color data is processed in accordance with the page description. When a device other than the target device is selected (a proofing device) the color data is intercepted, converted from the color space associated with the target device and converted to the color space associated with the proofing device by converting the data through a device independent color space.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD FOR PROCESSING A PAGE DESCRIPTION IN AN IMAGING SYSTEM

37 C.F.R. 1.71 AUTHORIZATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to imaging systems and processes and more specifically to systems and methods for processing a page description.

2. Description of the Related Art

Modern imaging-systems used in the pre-press, printing, computer graphics, and related industries are designed to operate using various different standards, sometimes called color spaces, for defining colors. These standards are conventionally categorized as either device dependent or device independent color spaces.

Device dependent color spaces, such as the Red, Blue, Green (RGB) and Cyan, Magenta, Yellow and Black (CMYK) color spaces generally produce different color results when using different output devices to form an image using the same input data. This variance occurs because device dependent color spaces produce colors which are dependent on the particular physical characteristics of the output device including the characteristics of the individual colorants and the characteristics of various combinations of the individual colorants.

Device independent color spaces specify color using standards which are not dependent on the physical characteristics of any particular output device. Thus, device independent color spaces such as CIE XYZ (1931) and CIE L*a*b* provide a way to specify a color in "absolute" terms and in a way that is related to human visual perception.

Conventionally, output devices used in imaging systems and particularly display devices printing devices and processes accept image data specified using a device dependent color space associated with the particular display devices printing device or printing process. For example, several conventional device dependent processes in particular are the SWOP® (CGA75 TR001-1995) process, DuPont Cromalin®, and 3M Matchprint®. Thus, when forming an image to be produced on an image-forming device using a particular device dependent color space, the image data is generated in accordance with the known color characteristics associated with the particular device dependent color space.

In some imaging systems, the image data is specified using a "page description" generated by a page make-up application A page description specifies the layout of a page and the colors of the graphical elements on the page expressed using a page description language suitable for interpretation in a computer by a raster image processor. Thus, one common component of an imaging system is a page description language interpreter, for instance as produced by Adobe Systems Incorporated for interpretation of the PostScript® page description language. One function of apparatus employing such interpreters is to accept, as input, imaging commands written in the page description language and to produce, as output, data signals compatible with an output device, such as a conventional dye-sublimation printer. Further pertinent background is presented in the POSTSCRIPT LANGUAGE REFERENCE MANUAL, SECOND EDITION, Adobe Systems Inc., pp. 176 - 200, 293-322 (Addison-Wesley 1990), the contents of which are incorporated herein by reference.

Because many imaging systems, and the printing industry in particular, use device-dependent color spaces, the target output device on which a page is to be printed or shown needs to be known at the time the page description is prepared and the colors are specified accordingly. Thus, most page descriptions specify color in terms of a device dependent color space determined by the color space associated with a target output device.

It is often desired to produce a "proof" of a page description before printing it on the target output device for which the page description was generated. A proof is a representation of the page produced using an output device different from that on which the final output is intended to be made (i.e. the target output device). Further, it

is often desirable to produce the proof on an output device designed to accept color data specified in a device dependent color space which is different from the color space of the target output device. Therefore, to produce a proof which accurately represents the color which will result from use of the target output device, the color data must be transformed from the device dependent color space of the target output device to the device dependent color space of the proofing system.

One conventional approach to proofing uses a mechanical proofing system wherein proofing inks are formulated to closely match those of the final printing process. This conventional approach requires a time consuming and complex matching process to be performed by a manufacturer and additionally is limited to the dye-sets provided by the manufacturer. Further, this conventional approach is not applicable to systems which do not use inks, such as computer displays.

Another approach to proofing requires that the page description describe the colors in device-independent form. Under this approach, when the page description is interpreted for generation using one type of output device, the device-independent colors are converted into amounts of colorant appropriate for that device (for example, the proofing system) and when printed on another device they are converted into different amounts of colorant suitable for the other device (for example, the target output device). However, this approach has the disadvantage that it requires a change in common working practice in that it requires page descriptions to specify color using a device independent color space. This approach has the further disadvantage that it requires the page makeup application be conversant with this method of page description.

Yet another conventional approach converts each pixel of the raster produced as a result of interpreting the page description using colors in the device-dependent form of the target system to the color required for the proofing system. However, this approach has the disadvantage that often a large amount of data has to be processed. A further disadvantage of this approach is that this approach is only suited for use with raster image data in continuous-tone form (i.e. not half-tone, which spatially distributes the information about the color). Additionally, to perform the color conversion on each pixel, all the color components of each pixel must be available at the same time, since the new color is a function of all the components of the original

color, yet it is often desirable to produce separate rasters of each color component individually and sequentially. Finally, this approach has the further drawback that information about the elements of the page is generally no longer available, so it is not possible to apply different conversions to different elements of the page using this approach.

Thus, there is a need for an improved imaging system having proofing capability that does not require a physical change of ink, which does not require a change in the conventional imaging (i.e. printing) work-flow, and does not require operation on a rasterized version of the page description and allows different elements of the page to have different conversions applied to them.

SUMMARY OF THE INVENTION

The system and method of the present invention generates a representation (i.e. a proof) of a page description prepared for a first output device or process by processing the page description to generate and produce the representation (proof) on a second output device or process. The first and second output devices or processes use different color spaces and thus the present invention performs a color space conversion.

In accordance with the present invention, the system and method receives a page description having color data specified in a first color space, detects the color space and responsive to the color space being device dependent, intercepts color data for conversion to the second color space (the proofing color space) associated with the output device to be used for producing the proof. Also in accordance with the present invention, the system and method attributes a device independent to meaning to intercepted color data as determined by the color characteristics of the output device for which the page description was prepared.

Also in accordance with the present invention, the color data included in the page description is converted to the proofing color space by first converting the color data to an intermediate device independent color space and by then converting the intermediate device independent color space data to the proofing color space.

Still further in accordance with the present invention, the conversion from the device dependent color space to the device independent color space and the

conversion from the device independent color space to the proofing color space (generally also a device dependent color space) are performed using a set of stored profiles.

Yet further in accordance with the present invention, the proofing mode of operation of the present invention is optionally selected. Proofing mode is selected by enabling color data interception and deselected by disabling color data interception. When proofing mode is selected, the page description is processed for production on the proofing output device and when proofing mode is deselected, the page description is processed for the output device for which the page description was prepared.

In another aspect of the present invention, the system includes an input module for providing a page description, memory for storing processing and data modules, a processor operatively coupled to the input module and to the memory for processing the page description using the stored processing and data modules and an output device for forming a physical manifestation of the processed page description (i.e. a print). The page description includes color data specified in either a device dependent color space or a device independent color space. The stored processing modules include a set up module for determining whether interception mode is selected, a raster image processing (RIP) subsystem for processing the page description. The RIP subsystem includes a detector for determining, during processing, whether color data is in device dependent or device independent form. The stored processing modules further include an interception module for processing device dependent color data detected when intercept mode is selected.

Also in accordance with this aspect of the present invention, the detector further includes an analyzer for identifying the device dependent color space.

Still further in accordance with this aspect of the present invention, the stored data modules include a plurality of stored data profiles for converting the color data from the identified color space to a second color space.

Yet further in accordance with this aspect of the present invention, the profiles for converting the color data from the identified color space to a second color space include profiles for converting the color data from the identified color space to a device

independent color space and profiles for converting data from the device independent color space to the color space associated with the output device.

In another aspect of the present invention, color data included in the page description is optionally specified by indicating the name of a specific colorant rather than by numerical values. Color data specified in this way ("named colors") is converted to the proofing color space by one of two ways, optionally selected. The first way includes converting named color to a device dependent numerical representation using conventional techniques, then converting the numerical representation to a device independent color space and then converting the device independent color space data to the proofing color space. The second way is by consulting a separate table of color names and if the color name is included in the table, substituting for that color the device independent color values associated with that name in the table and then by converting the device independent color space data to the proofing color space.

In yet another aspect of the present invention, color data is intercepted and is converted in accordance to the status of a set of intercept conditions.

Also in accordance with this aspect of the present invention, the intercept conditions include the type of the color space of the intercepted data, such as whether the color space is an RGB space or a CMYK space, or the type of element to which the intercepted data pertains, such as whether the data relates to a picture element.

In a further aspect of the present invention, color data which in conventional interpretation would overprint other colors which it overlays is constrained to continue to overprint where possible when converted to the proofing color space.

The features and advantages described in the specification are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of an image processing system in accordance with the present invention;

Figure 2 is a functional block diagram of the system illustrated in Figure 1;

Figure 3 is a flow diagram of a processing method in accordance with the present invention; and

Figure 4 is a flow diagram of an exemplary implementation of the invention using the PostScript® language.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 illustrates an image processing system **100** in accordance with the present invention. The major components of the system include a computer **101**, an input device **102** and an output device **103**. The major components of computer **101** include processor **104**, computer-readable memory **105**, storage device **106** and a computer bus **107** coupling memory **105** and storage device **106** to processor **104**. Memory **105** is conventional computer-readable memory suitable for storing program instructions and program data during the execution of a program by processor **104**. Storage device **106** is a conventional mass storage device for storing program instructions to be loaded into memory **105** during execution.

Computer **101** is a conventional computer such as a Macintosh personal computer manufactured by Apple Computers, Inc., a Pentium-based personal computer such as a personal computer manufactured by IBM, Inc. or a UNIX®-based workstation such as a SPARCStation manufactured by Sun Microsystems and the like. Alternatively, computer **101** may be a special purpose computing device having processing, storage and input and output capability.

Input device **102** is any device or machine which provides a page description in a "job" to be processed. Thus input device **102** may be a network connection, a floppy disk, a hard disk, including for example hard disk **106**, the output of another program or any other page description source and may be either internal or external to computer **101**. In this preferred embodiment, input device **102** is an input device external to computer **101**.

A page description is a specification of one or more pages including a specification of graphic elements and the colors of the elements to be included on each page. The page description is expressed in a page description language suitable for interpretation by an interpreter. In the preferred embodiment, the page description is provided using the PostScript® page description language specified in POSTSCRIPT LANGUAGE REFERENCE MANUAL, SECOND EDITION, Adobe Systems Inc., (Addison-Wesley 1990) by Adobe Systems Incorporated. The principles of the present invention, however are applicable to page descriptions expressed in other page description languages or by other specification means which specify colors in device dependent form.

Output devices **103** and **113** are devices suitable for forming a physical manifestation of the processed page description, such as a computer monitor (display), a laser printer, an ink jet printer, a photographic printer, a film recording device, a printing press or a dye sublimation printer and the like. In the preferred embodiment, output device **103** is a Seiko ColorPoint 2 835 RSH dye-sublimation printer and output device **113** is a film recording device from which printing plates for mounting on a SWOP® compliant printing press are prepared. Output devices **103** and **113** receive the processed page description in the form of data or signals and the like from computer **101**. Output devices **103** and **113** may each be coupled to computer **101** in a variety of ways including by a direct coupling, an indirect coupling such as through a network or by a physical transport of data using a portable storage medium. In certain modes of operation, output device **103** is used by system **100** as a proofing output device to produce a representation of a page description generated for another "target" output device (such as output device **113** also shown in Figure 1).

Storage Device **106** stores program modules **109-112** which are loaded into memory **105** for execution and processing by processor **104**. Although in the preferred embodiment, storage device **106** is a hard disk, storage device **106** may be any device suitable for storing program modules such as a compact disk read only memory (CD ROM), a cartridge drive or any other mass storage device located either internal or external to computer **101**.

Program Modules **109-112** include a RIP subsystem **109**, an interception module **110**, a set of profiles **111** and a setup module **112**.

The RIP subsystem **109** is a raster image processor (RIP) which includes instructions for interpreting the page description to generate a rasterized output compatible with output device **103**. RIP subsystem **109** processes the page description by reading characters from the incoming data included in the page description. When the characters form a recognizable token, RIP subsystem **109** responds by executing PostScript® language operators and values to process in accordance with the rules of the PostScript® language. RIP subsystem **109** includes a detector for determining, during processing, whether color data is in device dependent or device independent form.

In the preferred embodiment, set up module **112** is a set of PostScript® language instructions and is executed by processor **104** prior to processing a page description. Set up module **112** includes instructions for performing set up tasks to configure the RIP subsystem, prior to processing a page description, in accordance with various parameters selected by a user. In accordance with the present invention, one such parameter is an interception parameter selected by a user to either enable or disable a proofing mode. Proofing mode means that the page description is to be prepared for production, printing, display or other form of representation on an output device using a different color space from that for which the page description was prepared. If proofing mode is selected, set up module **112** enables interception of color data and if proofing mode is not selected, set up module **112** disables interception of color data. Set up module **112** also stores user-selected parameters identifying the target output device **113** and associated profiles for converting device dependent color to device independent color. Set up module **112** also stores user-selected parameters identifying the selected proofing output device **103** and an associated profile for converting from device independent colors to device dependent colors. The stored parameters are used to select the correct profiles for emulating the target output device **113** using the proofing output device **103**. In the preferred embodiment, other user-selected parameters controlled by set up module **112** include half-tone screening attributes, orientation, negative/positive selection, color and other controls including device specific controls. Further, in the preferred embodiment, a user alters the parameters and thus the configuration of RIP subsystem **109** using a

dialog box interface shown on a computer display coupled to computer **101** having selectable buttons, scrollable lists and text entry boxes and the like.

Interception module **110** includes PostScript® language instructions and extensions to the PostScript® language, in accordance with the present invention, for operating on and converting the format of color data selected for interception. When proofing mode is selected (interception is enabled), interception module **110** is used by RIP subsystem **109** to intercept device dependent color data included in a page description and convert the data to conform it with another selected color format. For example, when enabled, the interception module **110** converts intercepted color data to the device dependent color space associated with output device **103** (the proofing device).

Profiles **111** are color space conversion look-up tables for converting from a first color space to a second color space. Profiles **111** alternatively could be specified by color conversion equations, transforms or any other method or means of converting from a first color space to a second color space. Profiles **111** are used by interception module **110** for converting color data from a specified (characterized) device dependent color space to a device independent color space and for converting color data from a device independent color space to the device dependent color space associated with output device **103**. A characterized color space is one having substantially known color characteristics including color reproduction characteristics. Characterized color spaces include color spaces associated with specified printing and output processes, such as SWOP® (CGA75 TR001-1995) process, DuPont Cromalin®, and 3M Matchprint®, or with specific output device types such as a Sony Triniton Monitor, or a Seiko ColorPoint 2 835 RSH dye-sublimation printer or any other device or process having determined color reproduction characteristics.

Referring now to Figure 2, there is illustrated a functional block diagram showing the processing of color in one embodiment of imaging system **100**. The system **100** receives as input, a job **201** in the form of a page description. In accordance with the present invention the color data included in the job **201** is specified in either device dependent color or device independent color. In this embodiment, the colors assigned to the graphic elements described by the job **201** are specified in a device dependent format associated with a target output device (the original printer) **202**. In

accordance with the present invention system **100** has an interception feature which is selectively enabled or disabled, preferably under user control using a page set-up dialog box for example. When the interception feature is enabled, the colors specified in job **201** are intercepted by system **100** and processed for printing on a proofing output device **203**.

While a job **201** is provided to system **100**, the colors specified in job **201** are analyzed by a detector **204**. Detector **204** determines whether the colors given in the page description are specified in a device dependent color space. Detector **204** is operatively coupled to an interception enabler **206** functionally depicted as a switch in Figure 2. Interception enabler **206** also receives an interception enable /disable indication (data) **210** indicating whether the interception feature is enabled. In this embodiment, enabling the interception feature allows the job **201** to be printed on proofing output device **203** as opposed to the target output device **202**. With the interception feature enabled, the system further processes the page description to alter the color data in accordance with the device dependent color space associated with proofing output device **203**. Thus, with interception enabled, device dependent color data is intercepted and is converted to a device independent color space and processed as if specified in a device independent color space.

After being processed by detector **204**, colors of graphic elements in job **201** are processed in accordance with independent color adjustments specified in the page description in the job **201**. If the interception feature is enabled and the colors in the job **201** are specified in device dependent color space, then the colors in the job **201** are diverted by a switching means such as interception enabler **206** to alternate processing functions. In accordance with the alternate processing functions, a converter **208** receives color data from job **201** and uses one of the stored profiles (color space definitions) **209** to convert the color data of job **201** from device dependent data to device independent data. In accordance with the present invention, the selected profile **209** specifies the relationship between the device dependent color space associated with target output device **202** and a device independent color space such as CIE XYZ (1931). Converting the color data from device dependent form to the CIE XYZ (1931) color space attributes a device independent meaning to the color data as determined by the selected profile **209**.

The converted color data is next processed by a converter 211. Converter 211 uses a profile 213 to construct a PostScript® language color rendering dictionary which is used to convert the device independent data to the device dependent color space associated with proofing output device 203. Thus, intercepting device dependent color data (intended for a target output device) and converting that color data to a device independent form facilitates the further converting of the color data to a device dependent form suited for proofing output device 203. After converter 211 converts the color data, conventional adjuster 214 operates on the converted data to make calibration adjustments and other adjustments to compensate color for printer drift and other time-varying color factors. After adjuster 214 operates on the converted data, proofing output device 203 is used to generate a proof. The present system thus advantageously generates a proof having the colors which would have been generated had the print been produced on target output device 202 using the same page description (which was produced for the target output device 202 and that specified colors in the color space associated with target output device 202).

Figure 3 is a flow diagram of a method of processing a job in accordance with the present invention. The method initiates on receiving 300 a job including a page description. After receiving 300 a page description the method next retrieves 301 set-up information to configure the processing system to interpret the page description in accordance with parameters specified by a set up file (a PostScript® language fragment) associated with the job or as determined by the source of the job. Information specified in the set up file might include page set-up specifications such as page orientation information, color reversal information, intercept enablement information, printer selection information, and the like.

After retrieving 301 the job set-up parameters, the method next performs testing 302 to determine whether interception is enabled. If interception is not enabled, the method processes 303 the job in accordance with conventional page description processing methods. If, however, interception is enabled, the method constructs and analyzes 304 a PostScript® language dictionary (the interception dictionary) to prepare the method for interception and conversion of device dependent data. In a preferred embodiment, constructing and analyzing 304 includes retrieving a set of profiles 111 from hard disk 106. After constructing and analyzing 304 the dictionary, the method

processes 305 the job to provide a rasterized representation of the page description. Processing 305 continues until either the end of the page description is encountered (i.e. end of file) or until a graphics element is encountered which specifies color data. If the end of the page description is encountered, the method is done 306. If, however, during processing 305 a graphic element is recognized, the method proceeds to test 307. Test 307 determines whether the color data associated with the graphic element is device dependent or device independent data. If test 307 determines that the color data is device independent, the color data is conventionally processed 310 for the output device (i.e. proofing output device 203). If, however, test 307 determines that the color data is device dependent, the method converts 309 the data from the device dependent color space to a device independent color space using a selected profile. After conversion 309 of the color data to a device independent color space the method next converts 308 the color data from the device independent color space to the color space associated with the selected output printer. After the method performs conversion 308, the method conventionally processes 310 the color data for the selected output device and continues processing the page description.

The preferred embodiment additionally includes an enhanced overprinting feature. In accordance with the invention, overprinting characteristics are determined after color data is converted to the device dependent space associated with the output device. Overprinting is used in instances where at least one colorant is not needed to render a color. With overprinting, the area of the page occupied by that colorant is not painted at all, rather than being painted at the minimum possible value (typically 0), thereby allowing any of that colorant previously applied in the same area for an earlier graphical element to show through and combine with the other colorants which were painted. In certain cases, conversion to the device dependent color space associated with the output device causes colorants which were not previously required for rendering to now be required to reproduce the color correctly. However it is more advantageous to preserve overprinting characteristics in these circumstances than to reproduce the color accurately. In accordance with the present invention, overprinting is selectably preserved.

Overprinting is controlled by the PostScript Language's setoverprint operator. When overprinting is indicated, conventional behavior is to not render areas which

would have no colorant applied; when not overprinting (the normal mode of operation) absence of colorant means that the area would be rendered with no colorant, i.e. the background color. Conversion from one device color space to another may have introduced colorant where there was none before. In order to accommodate this, various artificially generated combinations of colorant having one or more colorants absent are converted using the same means as the colors provided by the page description. The absence of colorant after conversion is then determined by reference to the largest value (smallest in negative working) determined for each colorant from this sampling process rather than the simple absence of colorant.

Exemplary Implementation of the Preferred Embodiment Contained Within a PostScript® Language Compatible Interpreter

Figure 4 is a flow diagram of an exemplary implementation of the present invention that uses the PostScript® Level 2 language and the C programming language. The PostScript® Level 2 language provides for the expression of colors in various device dependent and device independent color spaces. The device dependent color spaces specified in the PostScript® Level 2 language are named DeviceCMYK, DeviceRGB and DeviceGray (collectively referred to herein as the "Device Color Spaces"). The present invention intercepts colors specified using a Device Color Space to convert them to the device independent color space (the CIE XYZ (1931)) used in PostScript® language-compatible interpreters to enable the Device Color Space colors to be processed in the same way that device independent colors are processed. The PostScript® language also provides for the expression of "named colors" in its Separation color space which can also give rise to colors in the Device Color Space when not using special purpose inks.

In this exemplary embodiment, interception mode is enabled and disabled by a user using a page set-up dialog box 401. Using page set-up dialog box 401, the system can be configured to handle up to four kinds of interception. The four kinds of interception are DeviceCMYKPicture, DeviceCMYKOther, DeviceRGB and NamedColor. When enabled, DeviceCMYKPicture, DeviceCMYKOther and DeviceRGB each intercept and modify color in accordance with the profile associated with the particular kind of interception. When enabled, NamedColors interception

intercepts any "named colors" specified in the PostScript® language Separation color space. DeviceCMYKPicture, DeviceCMYKOther, DeviceRGB and NamedColors each have an asscoiated set of intercept conditions (discussed below). For each kind of interception, when the intercept conditions are met and the interception kind is enabled, then the interception kind occurs.

Dialog box 401 is shown to a user on a computer display and has selectable buttons, scrollable lists, text entry boxes and the like to allow a user to specify printing configuration parameters including enabling and disabling of interception. The interception mode is preferably further controlled using dialog box 401 to allow a user to select printers or printing or proofing processes by name, one for each of three kinds of interception (DeviceCMYKPicture, DeviceCMYKOther and DeviceRGB) provided in this embodiment. In addition, the dialog box 401 allows a user to enable and disable the fourth kind of interception (NamedColors). The selection of a printer or printing or proofing process includes the selection of a target output device (for which the page description was prepared) as well as a selection of the proofing output device.

The parameters selected using dialog box 401 are used to generate a set-up file 402. Set-up file 402 includes the stored parameters which are used during configuration 404 to configure the system prior to interpreting 406 the page description. When a user enables interception (selects interception mode) using dialog box 401, set-up file 402 is generated to include a parameter for invoking the PostScript® language extension, in accordance with the present invention, which enables interception. Use of extensions to the PostScript® language is known in the art.

The setinterceptcolorspace extension is an operator that takes as its operand a PostScript® language dictionary. A dictionary is a table for associating pairs of PostScript® objects. The first object of a pair is called the key and the second object is an associated value. In operation, dictionary values are retrieved by performing a look-up in the table using the key. The setinterceptcolorspace operator takes as its operand a PostScript® language dictionary which has up to four keys: DeviceCMYKPicture, DeviceCMYKOther, DeviceRGB and NamedColor. DeviceCMYKPicture, DeviceCMYKOther, DeviceRGB and NamedColor are

PostScript® objects. The values associated with each key is an instance of a PostScript® language color space including extensions thereof.

When a PostScript® language page description requiring interpretation (a "job") is received **403** by the system, the system configures **404** the interpreter prior to interpreting **406** the page description. The interpreter configuration controls the way the job is interpreted. In particular, the interpreter configuration affects the subsequent interpretation and interception of device-dependent colors. This interpreter configuration is determined by executing (interpreting) the set-up file **402**, which includes invocation of the setinterceptcolorspace operator. The setinterceptcolorspace operator is a PostScript® language extension and has an operand dictionary including color space definitions retrieved from the stored profiles in response to the type of target and proofing output devices specified by the user. Advantageously, this means that the selection of a particular conversion does not need to be carried in or with the page description, but rather is specified by the setinterceptcolorspace operator before the job is interpreted.

Each key (DeviceCMYKPicture, DeviceCMYKOther, DeviceRGB and NamedColor) of the operand of setinterceptcolorspace operator has an associated color space definition in accordance with the configuration parameters selected using the dialog box **401**. The color space definitions are specified in the PostScript® language and extensions thereof. During configuration **404**, the system constructs **405** a dictionary. The dictionary includes a color space definition for up to four dictionary keys. Then the setinterceptcolorspace operator stores the contents of that dictionary so that they are available to the operators that are responsible for detection and interception later on. The setinterceptcolorspace operator analyzes each color space definition by extracting the color space definition from the dictionary and analyzing the color space definitions in the same way the standard PostScript® operator setcolorspace analyzes its color space operands.

In order to facilitate conversion from DeviceCMYK into device-independent color, and to provide tabular methods of expressing the conversion between color spaces, specifications for two new color spaces, which are extensions to the PostScript® language (the extensions) are also used in accordance with the present

invention. In this embodiment, the new color spaces are called CIETableABCD and CIETableABC.

When a particular key is not included in the dictionary operand of a call to setinterceptcolorspace, or its value is the PostScript® language value 'null', it means that no interception was selected for the particular key. However, when a particular key is included in the dictionary operand and it also has an associated color space definition, this means that interception is enabled for the particular key. Thus, when a page description is interpreted **406**, the type of color data associated with the key is intercepted **407**, provided that the other intercept triggering conditions (as described below) associated with the key are also satisfied. Advantageously, intercepted **407** color data is converted with respect to the color space definition as if that color data and color space definition had been specified explicitly using appropriate invocations of the standard PostScript® language operators setcolorspace and setcolor, image or colorimage. In this embodiment, the effects of the setinterceptcolorspace operator are subject to the standard PostScript® language mechanism of gsave and grestore.

The following describes the intercept triggering conditions for each key (for each kind of interception).

(i) The color space definition associated with the DeviceCMYKPicture key is applied only to graphic elements provided by the image and colorimage operators when the current color space is DeviceCMYK or DeviceGray, and only then when those graphic elements are not detected as special cases which are not thought to be pictures, namely those images which consist of large pixels or are only one pixel high or wide.

(ii) The color space definition associated with the DeviceCMYKOther key is applied to all other graphical elements when the current color space is DeviceCMYK or DeviceGray, including those images determined not to be pictures.

(iii) The color space definition associated with the DeviceRGB key is applied to all graphical elements when the current color space is DeviceRGB or when it is DeviceGray and the element has not already been subject to conversion according to one of the two DeviceCMYK rules above.

(iv) The color space definition associated with the NamedColor key is applied to all graphical elements when the current color space is Separation, and the name provided as part of the Separation color space is located in a table of named colors provided in the system for the purpose.

As a further disclosure of a preferred embodiment, exemplary source code of a computer software program, stored for instance on storage device **106** of computer **101** and for programming of processor **104**, is provided and described as follows:

Each PostScript® language operator is implemented as a C language function, many of which call other functions. Configuration **404** and construction and analysis **405** are implemented using the extension setinterceptcolorspace operator. The operator setinterceptcolorspace calls the C function setinterceptcolorspace_ implemented using C programming commands including:

```
/*****************************************************************/
int32 setinterceptcolorspace_ (void)
{
 OBJECT * theo;

 if (isEmpty (operandstack))
  return error_handler (STACKUNDERFLOW);

 theo = theTop (operandstack);

 if (theType (theITags (theo)) != ODICTIONARY)
  return error_handler (TYPECHECK);

 if (! dosetinterceptcolorspace (theo))
  return FALSE;

 pop (& operandstack);
 return TRUE;
}
/*****************************************************************/
```

The setinterceptcolorspace_ function calls the dosetinterceptcolorspace function to do detailed analysis of the dictionary operand of the setinterceptcolorspace operator, storing the results in the graphics state, implemented using C programming commands including:

```
/*****************************************************************/
int32 dosetinterceptcolorspace (OBJECT * theo)
{
 int32 result1, result2;
 OBJECT * interceptrgb, * interceptcmykpicture,
     * interceptcmykother, * interceptnamedcolor;
 COLOUR gscolor;

 /* scpy is a utility function to copy a given number of bytes */
 scpy ((uint8 *) & gscolor,
    (uint8 *) & theColour (gstate), sizeof (COLOUR));

 /* walk_dictionary is a utility function which iterates over every
   key/value pair in the PostScript® language dictionary given to it,
   calling (in this case) the function dictwalk_setintercept for
   each such pair, shown below */
 result1 = walk_dictionary (theo, dictwalk_setintercept, (void *) &
result2);

 interceptrgb =       thegsInterceptColorSpaceRGB (gstate);
 interceptcmykpicture = thegsInterceptColorSpaceCMYKPicture (gstate);
 interceptcmykother =  thegsInterceptColorSpaceCMYKOther (gstate);
 interceptnamedcolor =  thegsInterceptColorSpaceNamedColor (gstate);
 scpy ((uint8 *) & theColour (gstate),
    (uint8 *) & gscolor, sizeof (COLOUR));
 thegsInterceptColorSpaceRGB (gstate) =       interceptrgb;
 thegsInterceptColorSpaceCMYKPicture (gstate) = interceptcmykpicture;
 thegsInterceptColorSpaceCMYKOther (gstate) =  interceptcmykother;
 thegsInterceptColorSpaceNamedColor (gstate) =  interceptnamedcolor;

 thegsCurrentLoadedInterceptSpace (gstate) = NULL;
 thegsLoadedInterceptSpace (gstate) = SPACE_notset;

 return result1 && result2;
}
/*****************************************************************/
```

The dosetinterceptcolorspace function saves and restores parts of the graphics state which will be overwritten during analysis of the color spaces passed to setinterceptcolorspace and then calls a utility function walk_dictionary (not shown) to iterate over the key/value pairs contained in the dictionary, calling the function dictwalk_setintercept for each pair. The dictwalk_setintercept function is implemented using C programming commands including:

```
/*****************************************************************/
```

```
STATIC int32 dictwalk_setintercept (OBJECT * thekey,
                     OBJECT * theo,
                     void * presult)
{
 int32 name_id = NAME_null;
 int32 * result = (int32 *) presult;
 * result = TRUE;

 if (theName (theIValue (thekey)) ==
   system_names + NAME_DeviceRGB)
  name_id = NAME_DeviceRGB;
 else if (theName (theIValue (thekey)) ==
      system_names + NAME_DeviceCMYKPicture)
  name_id = NAME_DeviceCMYKPicture;
 else if (theName (theIValue (thekey)) ==
      system_names + NAME_DeviceCMYKOther)
  name_id = NAME_DeviceCMYKOther;
 else if (theName (theIValue (thekey)) ==
      system_names + NAME_NamedColor)
  name_id = NAME_NamedColor;

 if (name_id == NAME_null) {
  return TRUE;
 }

 if (theType (theITags (theo)) == ONULL) {
  theo = NULL;
 } else if (! determinecolorspace (theo)) {
  * result = FALSE;
  return FALSE;
 }

 switch (name_id) {
 case NAME_DeviceRGB:
  thegsInterceptColorSpaceRGB (gstate) = theo; break;
 case NAME_DeviceCMYKPicture:
  thegsInterceptColorSpaceCMYKPicture (gstate) = theo; break;
 case NAME_DeviceCMYKOther:
  thegsInterceptColorSpaceCMYKOther (gstate) = theo; break;
 case NAME_NamedColor:
  thegsInterceptColorSpaceNamedColor (gstate) = theo; break;
 default:
  HQFAIL ("if cascade above didnt work"); break;
 }

 return TRUE;
}
/*****************************************************************/
```

The dictwalk_setintercept function first determines which kind of intercept is represented by the key/value pair according to whether the name is DeviceRGB, DeviceCMYKPicture, DeviceCMYKOther or NamedColor and then calls a utility function determinecolorspace (not shown) to verify that the value does indeed represent a color space (and as a side-effect stores details of the color space in fields in the graphics state) and finally stores a reference to the value in a field in the graphic state appropriate to the kind of intercept. While interpreting a page description, colors presented to the interpreter by the setcmykcolor, setrgbcolor, sethsbcolor, setgray and setcolor operators are simply stored, also in the graphics state, until a painting operator (a function to draw a graphic element such as "fill" or "image") is presented to the interpreter. So, for example, the four numbers representing amounts of cyan, magenta, yellow and black given to the setcmykcolor operator are taken out of the structure representing the PostScript® language operand stack, and stored in the structure representing the graphics state. The setcmykcolor function is implemented using C programming commands including:

```
/*******************************************************************/
int32 setcmykcolor_ (void)
{
 register int32 i;
 SYSTEMVALUE args [4];
 int32 types [4];

 if (DEVICE_INVALID_CONTEXT ())
  return error_handler (UNDEFINED);

 /* Look for and extract four numbers from the operand stack using the
   utility function getN */
 if (! getN (args, types, 0, 4))
  return FALSE;

 for (i = 0; i < 4; i++) {
  if (args [i] < 0.0) {
   args [i] = 0.0;
  } else if (args [i] > 1.0) {
   args [i] = 1.0;
  }
  thegsInputColor (gstate) [i] = (USERVALUE)args [i];
 }
```

```
  thegsOutputColor (gstate) [0] = COLOR_NOT_SET;
  thegsColorSpace (gstate) = SPACE_DeviceCMYK;
  thegsRequiredRenderingIntent(gstate) = REPRO_TYPE_OTHER;
  theTags (thegsCurrentColorSpace (gstate)) = ONULL;
  thegsNoSetcolorYet (gstate) = FALSE;
  npop (4, & operandstack);
  return TRUE;
}
/******************************************************************/
```

When a painting operator is used by the job a corresponding function is called to process the graphic element. Functions which process a graphic object call a further function, lookup_color, which computes the color to be passed to the output printer by applying various color space transformations. This is done using the input color already stored in the graphics state as a result of execution of the setcmykcolor operator and similar, as described above, or in the case of the image and color image operators, this is done using input colors provided as data with the respective operator. The lookup_color function is implemented using C programming commands including:

```
/******************************************************************/
int32 lookup_color (USERVALUE input_color [4],
            COLORVALUE output_color [4],
            uint32 generation)
{
 uint8 mapped_space, xfer_applied;
 USERVALUE scratch_color[4] ;

 scratch_color[0] = input_color[0] ;
 scratch_color[1] = input_color[1] ;
 scratch_color[2] = input_color[2] ;
 scratch_color[3] = input_color[3] ;

 xfer_applied = FALSE;
 mapped_space = thegsColorSpace(gs:    );
 if ( ! map_input_space(&mapped_space, scratch_color, &xfer_applied) )
  return FALSE ;

 if ( generation != colorcache_generation )
  flush_colorcache(generation) ;

 return transform_color(mapped_space, scratch_color,
```

```
                thegsDeviceColorSpace(gstate), output_color,
                xfer_applied) ;
}
/*******************************************************************/
```

The lookup_color function first saves a local copy of the input color to be processed. Then it stores the current color space already in force (for example, DeviceCMYK if setcmykcolor has been used), as stored in the graphics state in the variable called mapped_space. Then the function map_input_space is called. The map_input_space function is implemented using C programming commands including:

```
/*******************************************************************/
int32 map_input_space(uint8 *input_space,
             USERVALUE output_color[4],
             uint8 *xfer_applied)
{
 OBJECT * cmykintercept;

 *xfer_applied = FALSE;

 switch (*input_space) {
 case SPACE_DeviceRGB:
  if (thegsInterceptColorSpaceRGB (gstate) != NULL) {
   LOADINTERCEPTSPACE (thegsInterceptColorSpaceRGB (gstate));
   *input_space = thegsLoadedInterceptSpace (gstate);
   switch (*input_space) {
   case SPACE_CIETable3:
   case SPACE_CIEBasedABC:
    break;
   default:
    return error_handler(RANGECHECK);
   }

   /* Call the utility function to apply the transfer function prior
     to the device-independent color transform being applied */
   if ( ! transform_color_transfer(output_color, output_color,
SPACE_DeviceRGB))
    return FALSE ;
   *xfer_applied = TRUE;
  }
  break;

 case SPACE_DeviceCMYK:
  if (thegsRequiredRenderingIntent(gstate) == REPRO_TYPE_PICTURE) {
```

```
      cmykintercept = thegsInterceptColorSpaceCMYKPicture (gstate);
    } else {
      cmykintercept = thegsInterceptColorSpaceCMYKOther (gstate);
    }

    if (cmykintercept != NULL) {
     LOADINTERCEPTSPACE (cmykintercept);
     *input_space = thegsLoadedInterceptSpace (gstate);
     switch (*input_space) {
     case SPACE_CIETable4:
      break;
     default:
      return error_handler(RANGECHECK);
     }

     if ( ! transform_color_transfer(output_color, output_color,
 SPACE_DeviceCMYK))
       return FALSE ;
     *xfer_applied = TRUE;
    }
    break;

   case SPACE_DeviceGray:
    if (thegsRequiredRenderingIntent(gstate) == REPRO_TYPE_PICTURE) {
     cmykintercept = thegsInterceptColorSpaceCMYKPicture (gstate);
    } else {
     cmykintercept = thegsInterceptColorSpaceCMYKOther (gstate);
    }

    if (cmykintercept != NULL) {
     LOADINTERCEPTSPACE (cmykintercept);
     *input_space = thegsLoadedInterceptSpace (gstate);
     switch (*input_space) {
     case SPACE_CIETable4:
      break;
     default:
      return error_handler(RANGECHECK);
     }

     if ( ! transform_color_transfer(output_color, output_color,
 SPACE_DeviceGray))
       return FALSE ;
     *xfer_applied = TRUE;
     output_color [3] = 1.0f - output_color [0]
     output_color [0] = output_color [1] = output_color [2] = 0.0f;
    } else if (thegsInterceptColorSpaceRGB (gstate) != NULL) {
     LOADINTERCEPTSPACE (thegsInterceptColorSpaceRGB (gstate));
     *input_space = thegsLoadedInterceptSpace (gstate);
```

```
    switch (*input_space) {
    case SPACE_CIETable3:
    case SPACE_CIEBasedABC:
      break;
    default:
      return error_handler(RANGECHECK);
    }

    if ( ! transform_color_transfer(output_color, output_color,
SPACE_DeviceGray))
      return FALSE ;
    *xfer_applied = TRUE;

    output_color [1] = output_color [2] = output_color [0];
  }
  break;

 default:
  /* color spaces other than DeviceCMYK, DeviceRGB and DeviceGray are not
    intercepted */
  break;
 }

 return TRUE;
}
/*****************************************************************/
```

If the current color space is being intercepted, then map_input_space causes mapped_space to be reassigned to one of the color spaces previously provided by the setinterceptcolorspace operator (an EalternateE color space). This means that the values in input_color are interpreted (by another function transform_color) with respect to the alternate color space rather than the one originally intended for the job, values used by the function in input_color to be interpreted with respect to that new space rather than the one originally intended.

The function transform_color repeatedly converts color values from one color space into another until it reaches the color space associated with the selected output device. Therefore by using the map_input_space function to change the starting color space, the color is diverted to a different color conversion path. The transform_color function is implemented using C programming commands including:

```
/*****************************************************************/
```

```
int32 transform_color (uint8 input_space, USERVALUE input_color[4],
              uint8 output_space, USERVALUE output_color[4],
              int32 xfer_applied)
{
 int32 i;
 uint8 dummy;

 USERVALUE k;
 OBJECT *theo;

 output_color [0] = input_color [0];
 output_color [1] = input_color [1];
 output_color [2] = input_color [2];
 output_color [3] = input_color [3];

 for (;;) {
  switch (input_space) {
  case SPACE_notset:
   return error_handler (UNREGISTERED);

  case SPACE_CIETable3:
  case SPACE_CIETable4:
   /* Do color conversions for tabular color conversions using utility
function */
    if (!interpolate_CIETable34_color(output_color, &input_space))
     return FALSE;
    continue;

  case SPACE_CIEBasedABC:
  case SPACE_CIEBasedA:
   /* Do color conversions for standard PostScript® language
     device-independent color spaces, and continue as above */
   if (! transform_color_cie_to_device (output_color, & input_space))
     return FALSE;
   continue;

  case SPACE_DeviceRGB:

   /* irrelevant code excised to convert DeviceRGB colors to output
     color space, if it is not already in the correct space. */
   break;

  case SPACE_DeviceCMYK:

   /* irrelevant code excised to convert DeviceCMYK colors to output
color space, if it is not already in the correct space. */
   break;
```

```
case SPACE_DeviceGray:

  /* irrelevant code excised to convert DeviceGray colors to output
    color space, if it is not already in the correct space. */
  break;

case SPACE_Pattern:

  /* irrelevant code excised to handle colors in pattern color space */
  break;

case SPACE_Indexed:

  /* irrelevant code excised to handle colors in pattern color space */
  continue;

case SPACE_Separation:
  if (thegsSpotColorIndex (gstate) == NONE_SEPARATION) {
    NARROW_01( output_color[0] );
    output_color [2] = output_color [3] = 0.0f;
    break;
  }
  else if (thegsNoSetcolorYet (gstate)) {
    output_color [0] = 0.0f;
    output_color [1] = output_color [2] = output_color [3] = 0.0f;
    input_space = SPACE_DeviceGray;
  } else {
    if (! transform_color_separation_to_base (output_color,
&input_space))
      return FALSE;
    if ((int32)input_space == SPACE_Separation) {
      /* We stayed as a spot color - no interception required */
      break;
    }
  }
  /* We converted to another color space - see if we need to
    intercept, and continue converting */
  if (!map_input_space(&input_space, output_color, &dummy))
    return FALSE ;
  continue;

 }

 default:
  return error_handler (UNREGISTERED);
 }
}
```

```
/* convert colors from their device color space form which they are
   now in, to appropriate colors for the device, including applying
   transfer function if that has not already been done, accounting for
   number of screen levels, calibration and range and scale of device
   codes */
 return TRUE;
}
/*****************************************************************/
```

Another way for device dependent color to arrive in a PostScript® job is using PostScript® is Separation color space. One possible outcome of interpreting 407 the PostScript® language's Separation color space is that the color so specified will be converted into a device-dependent color, and therefore transform_color also calls map_input_space in this case to intercept and divert the resultant color space.

Alternatively, when the named colors are being intercepted according to the NamedColors key in setinterceptcolorspace, the name of the color provided by the page description as part of the description of the Separation color space is looked up in a table provided with the system at the time the setcolorspace operator is called. If the name is found to be located in the table, the PostScript® language procedure which is provided by the page description to convert the named color into device dependent color is replaced by a procedure which instead provides a conversion from the named color to the device independent color stored in the table associated with that name. The procedure substituted also adjusts the device independent color to take account of any tint of the color indicated using normal PostScript® language methods by the page description. In this case, because the named color is converted to device independent color at the time when the page description interpreted normally would have converted to device dependent color, there is no further interception of the color in map_input_space.

The map_input_space function determines whether to change the input color space according to the kind (i.e. picture or not picture) of graphic element being processed (as previously set in the graphics state by the operator processing the graphic object), and the intercept color spaces (DeviceCMYKPicture, DeviceCMYKOther, and DeviceRGB) recorded in the graphic state by setinterceptcolorspace as described above, and the current color space (i.e. DeviceCMYK, DeviceRGB or DeviceGray). If interception 407 is required, it also has

the effect of applying PostScript® language transfer functions specified by the job so that the adjusted color which would have been sent to the originally intended output device is intercepted for conversion to the color space associated with the output proofing device.

Furthermore, when interception is required, map_input_space uses the macro LOADINTERCEPTSPACE to determine whether the data associated with the color space to be used is already stored in the graphics state. If the data is not in the graphics state, LOADINTERCEPTSPACE calls the function loadinterceptcolorspace to do so, which in turn calls determinecolorspace as above to carry out this task, and records that it has done so. The LOADINTERCEPTSPACE macro and the loadinterceptcolorspace funtion are implemented using C programming commands including:

```
/*******************************************************************/
#define LOADINTERCEPTSPACE(spaceo) MACRO_START \
 if (thegsCurrentLoadedInterceptSpace (gstate) == NULL || \
  thegsCurrentLoadedInterceptSpace (gstate)->_d0.transfer != \
   (spaceo)->_d0.transfer || \
  thegsCurrentLoadedInterceptSpace (gstate)->_d1.transfer != \
   (spaceo)->_d1.transfer) \
   if (! loadinterceptcolorspace (spaceo)) return FALSE; \
MACRO_END


int32 loadinterceptcolorspace (OBJECT * thespace)
{
 uint8 space, requiredrenderingintent;
 OBJECT current_space;
 USERVALUE input_color [4];
 COLORVALUE output_color [4];
 OBJECT *named_colour_intercept_space;

 space = thegsColorSpace (gstate);
 OCopy (current_space, thegsCurrentColorSpace (gstate));
 input_color [0] = thegsInputColor (gstate) [0];
 input_color [1] = thegsInputColor (gstate) [1];
 input_color [2] = thegsInputColor (gstate) [2];
 input_color [3] = thegsInputColor (gstate) [3];
 output_color [0] = thegsOutputColor (gstate) [0];
 output_color [1] = thegsOutputColor (gstate) [1];
 output_color [2] = thegsOutputColor (gstate) [2];
```

```
output_color [3] = thegsOutputColor (gstate) [3];
requiredrenderingintent = thegsRequiredRenderingIntent (gstate);

/* detemrinecolorspace is a utility function for analyzing color
   spaces and extracting contents and placing them in the
   graphics state */
if (! determinecolorspace (thespace))
  return FALSE;

thegsCurrentLoadedInterceptSpace (gstate) = thespace;
thegsLoadedInterceptSpace (gstate) = thegsColorSpace (gstate);
OCopy (thegsCurrentColorSpace (gstate), current_space);
thegsColorSpace (gstate) = (uint8) space;
thegsInputColor (gstate) [0] = input_color [0];
thegsInputColor (gstate) [1] = input_color [1];
thegsInputColor (gstate) [2] = input_color [2];
thegsInputColor (gstate) [3] = input_color [3];
thegsOutputColor (gstate) [0] = output_color [0];
thegsOutputColor (gstate) [1] = output_color [1];
thegsOutputColor (gstate) [2] = output_color [2];
thegsOutputColor (gstate) [3] = output_color [3];
thegsRequiredRenderingIntent (gstate) = requiredrenderingintent;

return TRUE;
}
/*****************************************************************/
```

The function overprint_colors is used to determine the color components against which to compare a color in order to determine whether we should overprint a color component (after transformations to it are complete). Normally overprinting would be done if the overprinting is turned on (tested elsewhere) and a color component is zero (or its maximum value in negative printing). When intercepting colors, however, a set of sample DeviceCMYK values are converted according to the kind of interception being used and the result of the conversion used to compare against.

The overprint_colors funtion is implemented using C programming commands including:

```
/* ------------------------------------------------------------ */
void overprint_colors (COLORVALUE overprint [4], COLORVALUE * pMaxBlack,
            int32 * pIntercepted)
{
 static COLORVALUE theoverprint [4];
```

```
static COLORVALUE themaxblack;
static int32 was_intercepted;
static uint32 overprint_generation = 0;

USERVALUE in [4];
COLORVALUE out [4];
uint8 saved_space;
uint8 result_space; /* dummy, unused */
OBJECT * cmykintercept = NULL;

if (overprint_generation != thegsColorSpaceNo (gstate)) {
 overprint_generation = thegsColorSpaceNo (gstate);

 was_intercepted = FALSE;
 /* decide what CMYK intercept to use */
 if (thegsRequiredRenderingIntent(gstate) == REPRO_TYPE_PICTURE) {
  cmykintercept = thegsInterceptColorSpaceCMYKPicture (gstate);
 } else {
  cmykintercept = thegsInterceptColorSpaceCMYKOther (gstate);
 }
 was_intercepted = cmykintercept != NULL;

 if (was_intercepted) {
  saved_space = thegsColorSpace (gstate);
  thegsColorSpace (gstate) = SPACE_DeviceCMYK;

  in [0] = 0.0f; in [1] = 1.0f; in [2] = 1.0f; in [3] = 0.0f;
  lookup_color (in, out, thegsColorSpaceNo(gstate),
             &result_space);
  theoverprint [0] = out [0];
  theoverprint [3] = out [3];

  in [0] = 1.0f; in [1] = 0.0f;
  lookup_color (in, out, thegsColorSpaceNo(gstate),
             &result_space);
  theoverprint [1] = out [1];
  if (erasenegative [3] ?
     theoverprint [3] < out [3] :
     theoverprint [3] > out [3])
   theoverprint [3] = out [3];

  in [1] = 1.0f; in [2] = 0.0f;
  lookup_color (in, out, thegsColorSpaceNo(gstate),
             &result_space);
  theoverprint [2] = out [2];
  if (erasenegative [3] ?
     theoverprint [3] < out [3] :
     theoverprint [3] > out [3])
```

```
      theoverprint [3] = out [3];

      in [0] = in [1] = in [2] = 0.0f; in [3] = 1.0f;
      lookup_color (in, out, thegsColorSpaceNo(gstate),
                    &result_space);
      themaxblack = out [3];

      thegsColorSpace (gstate) = saved_space;
    } else {
      /* we are using raw cmyk */
      theoverprint [0] = (COLORVALUE) (erasenegative [0] ? 0 : halfdots
  [0]);
      theoverprint [1] = (COLORVALUE) (erasenegative [1] ? 0 : halfdots
  [1]);
      theoverprint [2] = (COLORVALUE) (erasenegative [2] ? 0 : halfdots
  [2]);
      theoverprint [3] = (COLORVALUE) (erasenegative [3] ? 0 : halfdots
  [3]);
      themaxblack = (COLORVALUE) (halfdots [3] - theoverprint [3]);
    }
  }

  overprint [0] = theoverprint [0];
  overprint [1] = theoverprint [1];
  overprint [2] = theoverprint [2];
  overprint [3] = theoverprint [3];
  * pMaxBlack = themaxblack;
  * pIntercepted = was_intercepted;
}
/* ------------------------------------------------------------ */
```

While interpreting Separation color spaces, if interception of named colors is enabled, the name is looked up in a table (using a PostScript® procedure NCknowncolorname) and if present the so-called tint transform procedure of the Separation color space is substituted (also by NCknowncolorname) by one which yields the device independent equivalent, before that color space definition is conventionally processed.

This is done by the C function determineseparationspace (which is a subordinate function of determinecolorspace, mentioned above). That part of determineseparationspace related to intercepting named colors includes:

```
/* ------------------------------------------------------------ */
```

```
/* see if this separation is one we're intercepting */
if (thegsInterceptColorSpaceNamedColor(gstate) != NULL)
{
  int32 stackcount = theStackSize(operandstack);
  if (! push(&thegsSepName(gstate), &operandstack))
    return FALSE;

  /* Determine (by calling out to PostScript® language) whether
    the named color is in the table of named colors */
  if (run_ps_string((uint8*)"/HqnNamedColor /ProcSet findresource
/NCknowncolorname get exec"))
  {
    OBJECT *result;
    if (theStackSize(operandstack) <= stackcount)
      return error_handler(STACKUNDERFLOW);
    result = theTop (operandstack) ;
    tags = theITags (result);
    if (theType (tags) != OBOOLEAN)
      return error_handler(TYPECHECK);
    if ((theBool(theIValue(result))))
    {
      if (theStackSize(operandstack) != stackcount + 3)
        return error_handler(UNDEFINED);
      OCopy(thegsSepTintTransform(gstate),
theOList(theIValue(thegsInterceptColorSpaceNamedColor(gstate)))[3]);

      LOADINTERCEPTSPACE(&(theOList(theIValue(thegsInterceptColor
SpaceNamedColor(gstate)))) [2]);
      thegsBaseColorSpace(gstate) =
thegsLoadedInterceptSpace(gstate);
      thegsRequiredRenderingIntent(gstate) = REPRO_TYPE_NAMEDCOLOR;

      npop(3, &operandstack);
    } else {
      if (theStackSize(operandstack) != stackcount + 1)
        return error_handler(UNDEFINED);
      npop(1, &operandstack);
    }
  } else {
    return error_handler(UNDEFINED);
  }
/* ---------------------------------------------------------- */
```

The PostScript® language procedure NCknowncolorname referred to above is implemented using a collection of PostScript® language commands which determine from the name of the color to be looked up the name of a PostScript® language

resource file containing part of the table of named colors, examines that table and if the color is present substitutes a tint transform procedure which applies any PostScript® language tint value to the looked up device independent value of the color.

The PostScript® language commands for NCknowncolorname and related procedures include:

```
/NCdict 10 dict def
% array with lower -> upper case
/a (a) 0 get def
/z (z) 0 get def
/A (A) 0 get def
/NCcasemap 256 string def
0 1 255 { NCcasemap exch dup put } for
//a 1 //z { NCcasemap exch dup //a sub //A add put } for

/NCstr 256 string def

% converts a string to upper case; converts name to string too, if
necessary
/NCupcase
{
 //NCstr cvs
 dup length 1 sub 0 1 3 -1 roll
 { 2 copy get //NCcasemap exch get 2 index 3 1 roll put }
 for
}
bind def


/NCresourcename
{
 (PAN) anchorsearch
 {
  pop (CVU) search
  { pop pop pop /PanU true }
  {
   (CV) search
   { pop pop pop /PanV true }
   { pop false }
   ifelse
  }
  ifelse
 }
 { pop false }
 ifelse
```

```
}
bind def

/NCknowncolorname
{
 //NCupcase exec
 dup //NCresourcename exec
 {
  /NamedColor findresource
  exch
  2 copy known
  { true }
  { pop pop false }
  ifelse
 }
 % no resource matching that spec
 { pop false }
 ifelse
}
bind def


/NCcolorspace /XYZ-D50 /ColorSpace findresource def

//NCcolorspace 1 get /WhitePoint get aload pop
/Zn exch def
/Yn exch def
/Xn exch def

/NCtinttransform
{
 currentcolorspace 1 get //NCknowncolorname exec
 {
  get exec
  3 index 1 ne
  {
   3 index 0 eq {
    pop pop pop pop
    //Xn //Yn //Zn
   }{
    //NCdict begin
     /Z exch def
     /Y exch def
     /X exch def
     /T exch def
     /Xf X //Xn div 0.008856 le { X //Xn div 7.787 mul 16 116 div add } {
X //Xn div 1 3 div exp } ifelse def
```

```
    /Yf Y //Yn div 0.008856 le { Y //Yn div 7.787 mul 16 116 div add } {
Y //Yn div 1 3 div exp } ifelse def
    /Zf Z //Zn div 0.008856 le { Z //Zn div 7.787 mul 16 116 div add } {
Z //Zn div 1 3 div exp } ifelse def
    /L* Yf 116 mul 16 sub def
    /a* Xf Yf sub 500 mul def
    /b* Yf Zf sub 200 mul def
    /L* T L* 100 sub mul 100 add def
    /Yf L* 8 gt { L* 16 add 116 div } { L* 903.3 div 7.787 mul 16 116
div add } ifelse def
    /Xf a* 500 div Yf add def
    /Zf Yf b* 200 div sub def
    /X Xf 0.2069 gt { Xf 3 exp //Xn mul } { Xf 16 116 div sub 7.787 div
//Xn mul } ifelse def
    /Y L* 8     gt { Yf 3 exp //Yn mul } { Yf 16 116 div sub 7.787 div
//Yn mul } ifelse def
    /Z Zf 0.2069 gt { Zf 3 exp //Zn mul } { Zf 16 116 div sub 7.787 div
//Zn mul } ifelse def
    X Y Z
   end
  } ifelse
 }{
  4 -1 roll pop
 } ifelse
}
{
 % unknown colour - use the default tint transform
 currentcolorspace 3 get exec
}
ifelse
}
bind def
/**********************************************************************/
```

Thus, the present system and method advantageously provides a means of processing a page description prepared for a target output device and system and generates a physical manifestation (a proof) of the page description using an output device which has color space specifications different from the target output device or system and yet provides an approximation of the result which will occur if the same page description is processed on the target output device or system. As particular advantage in the proofing context, a single page description can be used for both the proofing device and the targeted output device.

The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

CLAIMS

What is claimed is:

1. A method of processing a page description having color information specified in a first color space for output to an output device having a second color space, comprising:

associating the first color space with a profile;

determining whether the first color space is a device dependent color space;

responsive to the color space being a device dependent color space, converting the color information from the first color space to the second color space responsive to the association; and

processing the page description using the converted color information, for output to the output device.

2. The method of claim 1 wherein associating the first color space with a profile includes selecting a profile from a set of profiles.

3. The method of claim 2 wherein the set of profiles include a profile corresponding to any of a printing process, a proofing process or a device capable of reproducing color pages.

4. The method of claim 1 wherein converting the color information from the first color space to the second color space responsive to the association comprises converting the color information from the first color space to a second color space using a device independent color space.

5. The method of claim 1 wherein converting the color information from the first color space to the second color space responsive to the association comprises:

converting the color information from the first color space to a device independent color space responsive to the association; and

converting the converted color information from the device independent color space to the second color space.

6. The method of claim 5 wherein converting the color information from the first color space to a device independent color space responsive to the association comprises providing a table for converting the color information from the first color space to the device independent color space .

7. The method of claim 1 wherein converting the color information from the first color space to the second color space responsive to the association comprises providing a table for converting the color information from the first color space to a second color space.

8. The method of claim 7 wherein the table is derived using a device independent color space.

9. A method of processing a page description having color information specified in a color space associated with a first output device, for output to a second output device, comprising:

determining the color space associated with the first output device;

selectively intercepting color information;

converting the intercepted color information to a color space associated with the second output device responsive to the determined color space; and

processing the page description using the converted color information, for output to the second output device.

10. The method of claim 9 wherein determining the color space associated with the first output device includes selecting a color space from a set of profiles including a profile corresponding to any of a printing process, a proofing process or a device capable of reproducing color pages.

11. The method of claim 9 wherein selectively intercepting color information includes determining the status of selected intercept conditions; and wherein converting the intercepted color information to a color space associated with the

second device responsive to the determined color space is further responsive to the status of the selected intercept conditions.

12. The method of claim 11 wherein the intercept conditions include whether the first color space is RGB.

13. The method of claim 11 wherein the intercept conditions include whether the first color space is CMYK.

14. The method of claim 11 wherein the intercept conditions include whether the first color space is CMYK and whether the color information relates to a picture.

15. The method of claim 11 wherein the intercept conditions include whether the intercepted color information is a color specified by a name.

16. The method of claim 9 wherein converting the intercepted color information to a color space associated with the second output device responsive to the determined color space includes using a device independent color space.

17. The method of claim 9 wherein converting the intercepted color information to a color space associated with the second output device responsive to the determined color space comprises:

converting the color information from the color space associated with the input device to a device independent color space responsive to the determined color space; and

converting the converted color information from the device independent color space to the color space associated with the output device.

18. The method of claim 17 wherein converting the color information from the color space associated with the input device to a device independent color space responsive to the determined color space comprises providing a table for converting

the color information from the color space associated with the input device to a device independent color space.

19. The method of claim 9 wherein converting the intercepted color information to a color space associated with the second output device responsive to the determined color space includes providing a table for converting the color information from the color space associated with the first output device to the color space associated with the second output device.

20. The method of claim 19 wherein the table is derived using a device independent color space.

21. A system for producing a rasterized representation of a page description , comprising:

an input subsystem configured to receive the page description having color information specified in a color space associated with a first output device;

an interception subsystem operatively coupled to the input subsystem configured to determine the color space associated with the first output device, to selectively intercept color information, to convert the intercepted color information to a color space associated with the second output device responsive to the determined color space; and

a processing subsystem operatively coupled to the interception subsystem configured to process the page description using the converted color information, and further configured to rasterize information included in the page description for output to the second output device.

22. The system of claim 21 wherein the interception subsystem further includes a profile associating color information associated with a device dependent color space to color information associated with a device independent color space.

23. A computer-readable storage device for use in an imaging system, the storage device including data for implementing a method comprising the steps of:

receiving a page description having color information specified in a color space associated with the first output device;

determining the color space associated with the first output device;

selectively intercepting color information;

converting the intercepted color information to a color space associated with a second output device responsive to the determined color space; and

processing the page description using the converted color information, for output to the second output device.

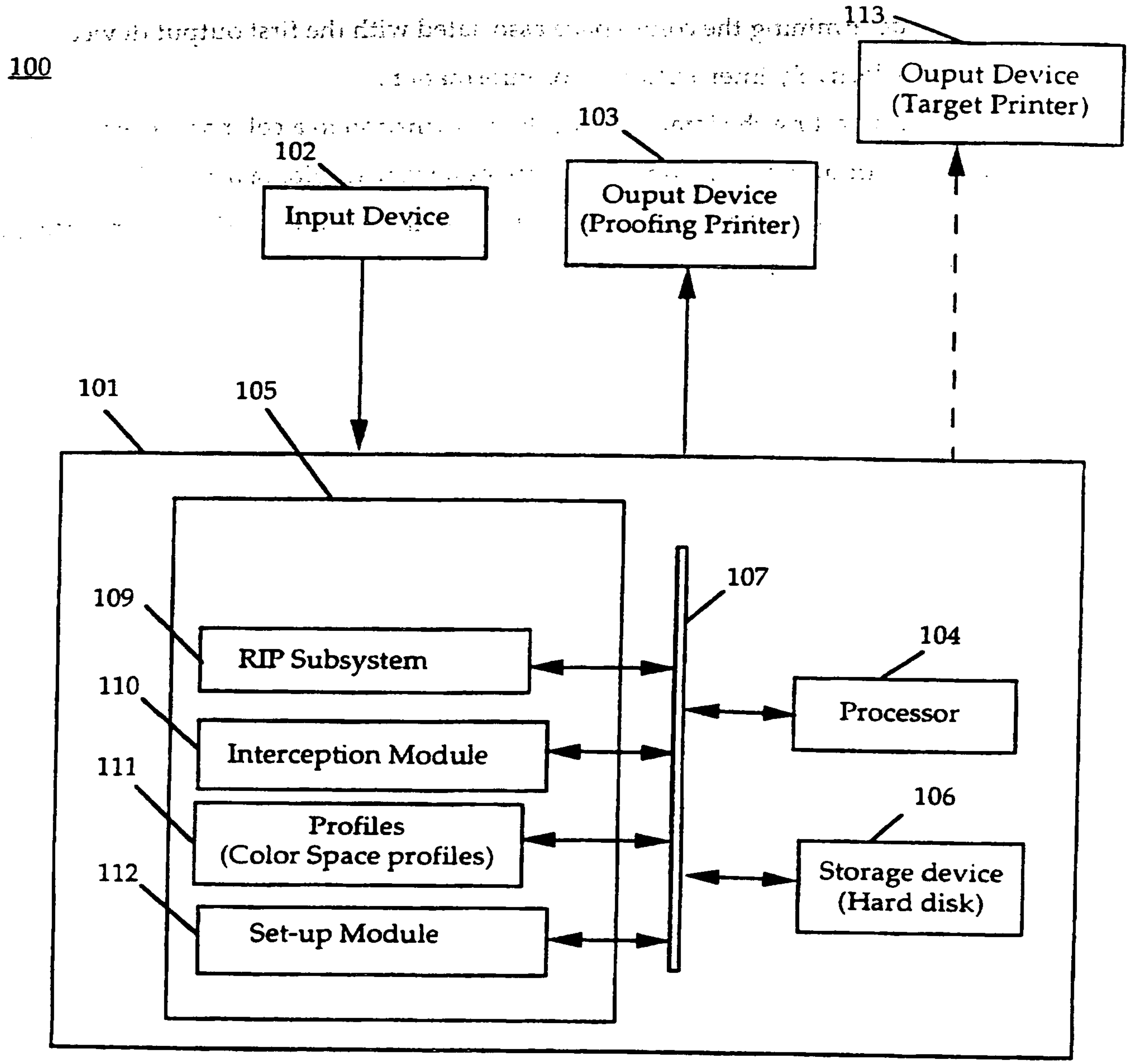

100
113
Ouput Device
(Target Printer)
103
Ouput Device
(Proofing Printer)
102
Input Device
101
105
109
RIP Subsystem
110
Interception Module
111
Profiles
(Color Space profiles)
112
Set-up Module
107
104
Processor
106
Storage device
(Hard disk)

Figure 1

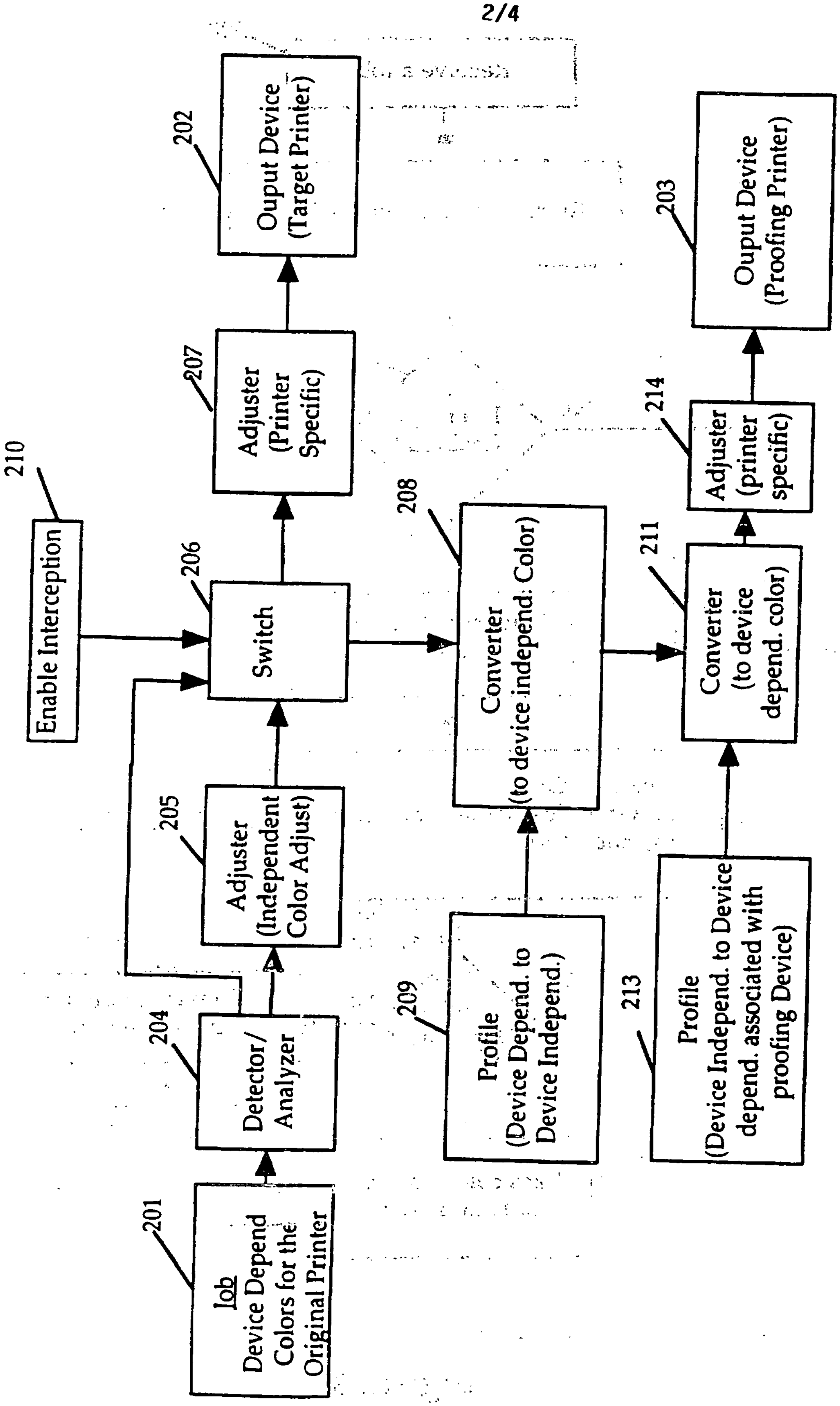

201
Job
Device Depend
Colors for the
Original Printer
204
Detector/
Analyzer
205
Adjuster
(Independent
Color Adjust)
206
Switch
210
Enable Interception
207
Adjuster
(Printer
Specific)
202
Ouput Device
(Target Printer)
208
Converter
(to device independ: Color)
209
Profile
(Device Depend. to
Device Independ.)
211
Converter
(to device
depend. color)
213
Profile
(Device Independ. to Device
depend. associated with
proofing Device)
214
Adjuster
(printer
specific)
203
Ouput Device
(Proofing Printer)

Figure 2

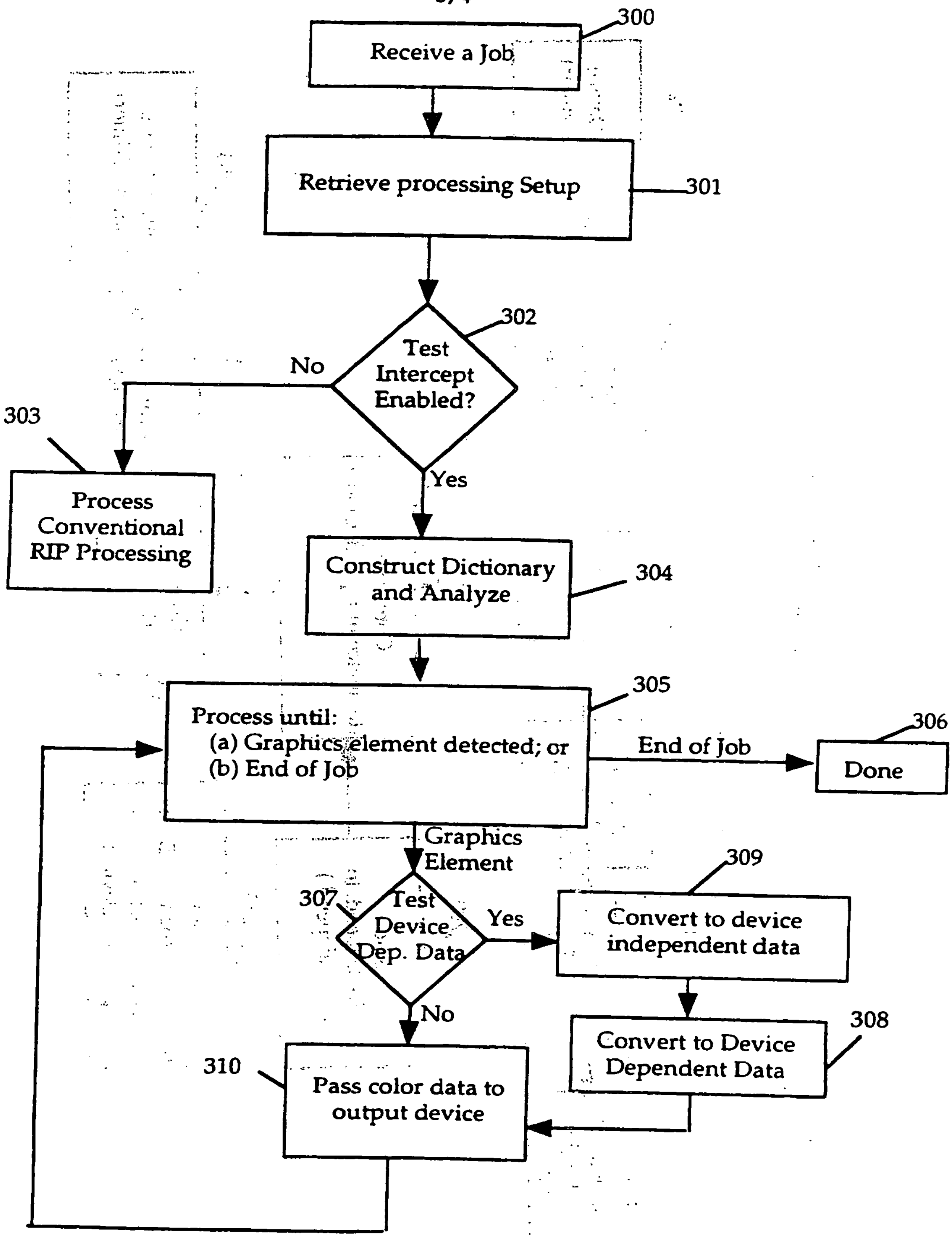

Receive a Job
300
Retrieve processing Setup
301
Test Intercept Enabled?
302
No
Yes
303
Process Conventional RIP Processing
Construct Dictionary and Analyze
304
Process until:
(a) Graphics element detected; or
(b) End of Job
305
End of Job
Done
306
Graphics Element
Test Device Dep. Data
307
Yes
No
Convert to device independent data
309
Convert to Device Dependent Data
308
Pass color data to output device
310

Figure 3

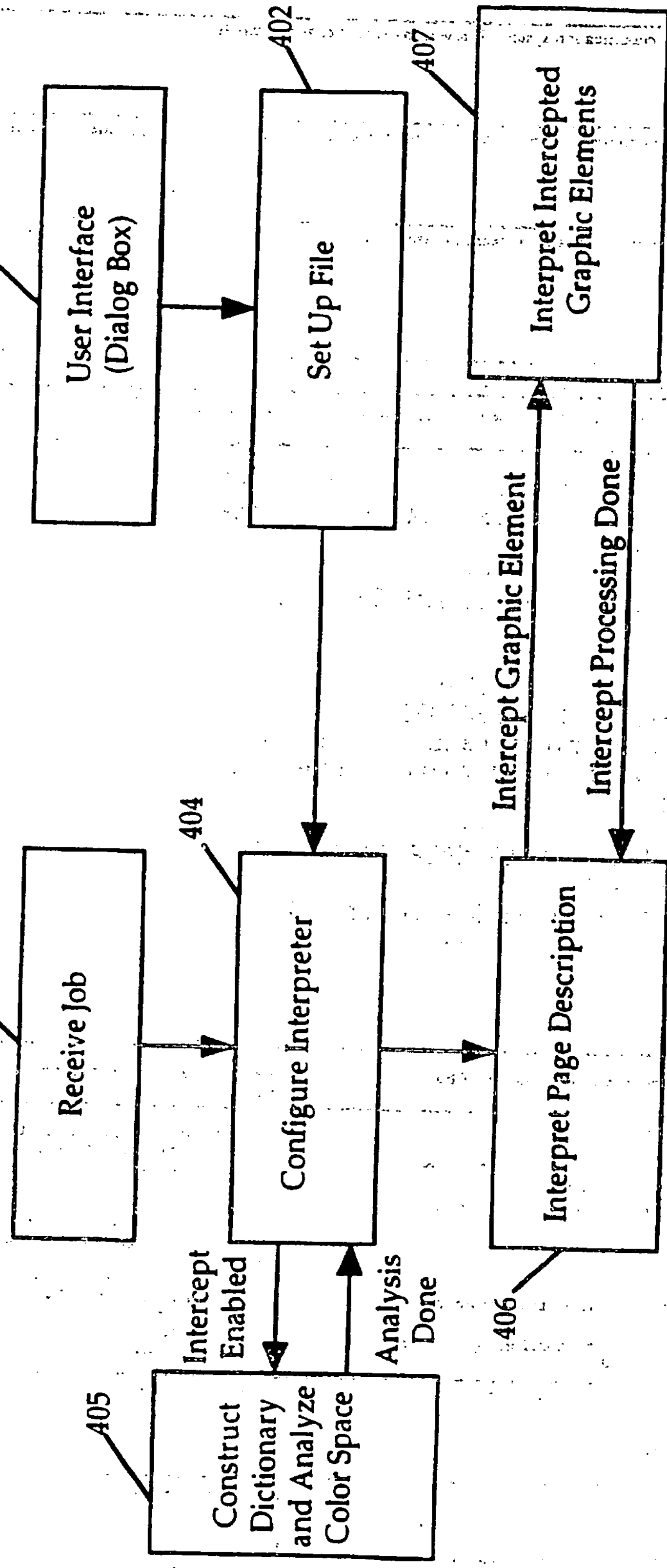

401
User Interface (Dialog Box)
402
Set Up File
407
Interpret Intercepted Graphic Elements
403
Receive Job
404
Configure Interpreter
406
Interpret Page Description
Intercept Graphic Element
Intercept Processing Done
405
Construct Dictionary and Analyze Color Space
Intercept Enabled
Analysis Done

Figure 4

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP 97/02229

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06T11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 706 285 A (CANON KK) 10 April 1996 see column 2, line 15 - column 3, line 3; figures 2,4 ---	1-23
A	MACDONALD L W: "DEVELOPMENTS IN COLOUR MANAGEMENT SYSTEMS" DISPLAYS, vol. 16, no. 4, May 1996, pages 203-211, XP000607335 see page 205, right-hand column, line 1 - page 208, right-hand column, line 16; figures 1-4 ---	1-23
A	US 5 243 414 A (DALRYMPLE JOHN C ET AL) 7 September 1993 see column 12, line 6 - column 14 --- -/--	2

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
27 August 1997	05.09.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Perez Molina, E

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inte. ...nal Application No
PCT/EP 97/02229

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4 941 038 A (WALOWIT ERIC) 10 July 1990 see column 2, line 41 - line 58 -----	1
A	EP 0 700 198 A (XEROX CORP) 6 March 1996 -----	

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/EP 97/02229

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0706285 A	10-04-96	JP 8098040 A	12-04-96
		JP 8090838 A	09-04-96
US 5243414 A	07-09-93	JP 5199410 A	06-08-93
US 4941038 A	10-07-90	US 4965672 A	23-10-90
		CN 1044544 A	08-08-90
		EP 0378448 A	18-07-90
		JP 2289367 A	29-11-90
		DE 3850702 D	25-08-94
		DE 3850702 T	27-10-94
		EP 0291300 A	17-11-88
		JP 1020773 A	24-01-89
EP 0700198 A	06-03-96	JP 8077341 A	22-03-96
		US 5581376 A	03-12-96

This Page Blank (uspto)